Program Structures & Algorithms

INFO 6205 Fall 2019

Assignment 3

1. Union-find Code

In the union-find assignment, I completed the code UF\_HWQUPC with the function find(), mergeComponents() and doPathCompression(). Then I wrote the function count() for connecting each node to other nodes by using uf.union(), and for each time make the parameter result adding one. Finally, for the main part, I set n to 10 / 100 / 1000 /10000 to test the average number of processing for different amount of node n and compare the average value to the output of 0.5\*n\*ln(n). The cases of union-find assignment led us think extensively about improving the time complexity.

1. Run-Time Graph & Conclusion

图片包含 天线, 物体

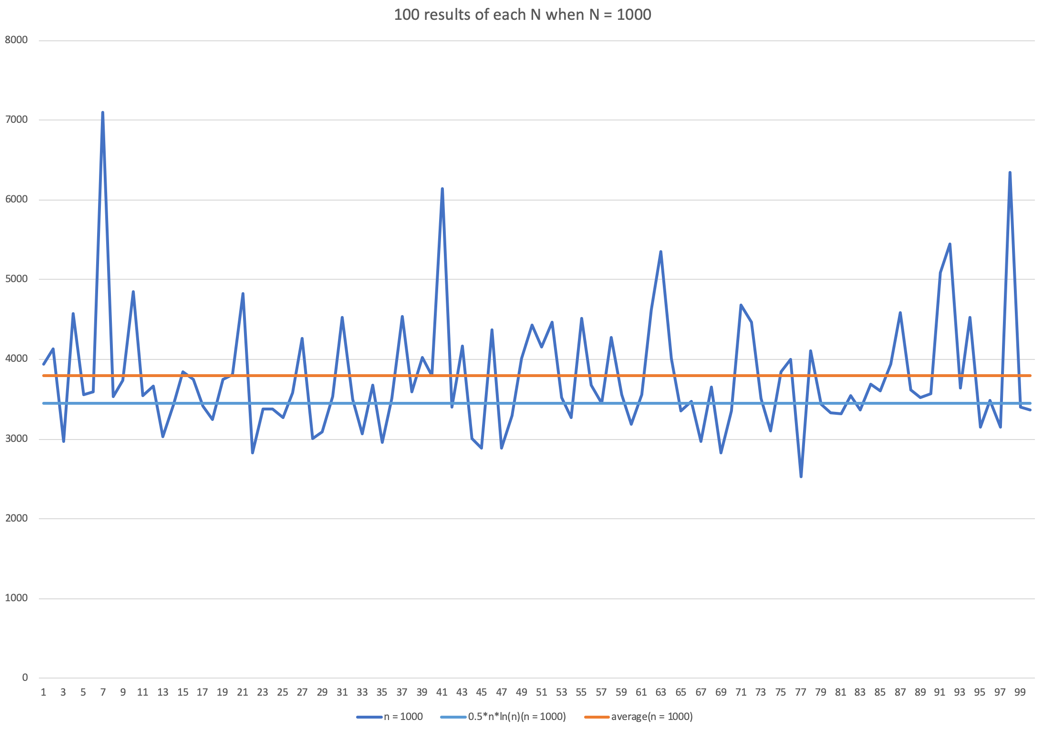
描述已自动生成

Graph When N = 10

图片包含 天线, 物体

描述已自动生成

Graph When N = 100



Graph When N = 1000

图片包含 物体, 天线, 屏幕截图

描述已自动生成

Graph When N = 10000

As the conclusion may not be clarified directly through the graph, I draw the result for 100 times and calculate the average, all the four results are shown on these screenshots. And all the data is saved in the extended excel file.

图片包含 屏幕截图

描述已自动生成

Code Output

From the screenshots, we can see that, the curve for 0.5\*n\*ln(n) is always lower than the curve for the average value, to get the exact difference, we also need to calculate the difference between 0.5\*n\*ln(n) and the result based on the current result. The following graph includes all these four conditions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | 1/2nln(n) | WQUPC | difference | alteration |
| 10 | 11.51 | 17.29 | 5.78 | 50.21% |
| 100 | 230.26 | 255.56 | 25.3 | 10.99% |
| 1000 | 3453.88 | 3789.96 | 336.08 | 9.73% |
| 10000 | 46051.7 | 48519.4 | 2467.7 | 5.36% |

Difference Table

From this table we can find that with the growing of n from 10 to 10000, the difference between the 0.5\*n\*ln(n) and the WQUPC result is shrinking, which prove that the hypothesis is almost right, after n-1 times generating random pairs of integers upon n numbers, to connect all the sites together, we need 0.5\*n\*ln(n) pairs generated to accomplish this.